Introduction to Digital Speech Processing Homework 3 FAQ

SRILM Compilation

Q1. 安裝說明

i686:

- 解壓縮 srilm-1.5.10.tar.gz
- 修改 **Makefile**, 在檔案上端加入 SRILM 變數到你的 srilm 目錄(絕對路徑)
  - ex: # SRILM = /home/speech/stolcke/project/srilm/devel
    => SRILM = /root/srilm-1.5.10

    更多請詳見 Q3
    以下 SRILM 路徑都以此路徑舉例說明
- 修改檔案 **common/Makefile.machine.i686**
  - 17 行的 CC 改成 **CC = /usr/bin/gcc $(GCC_FLAGS)**
  - 18 行的 CXX 改成 **CXX = /usr/bin/g++ $(GCC_FLAGS) -DINSTANTIATE_TEMPLATES**
  - 52, 53 行的 tcl 設定改成下面三行
    NO_TCL = X
    TCL_INCLUDE =
    TCL_LIBRARY =
- 請確定 sbin/ 資料夾底下的程式為可執行檔。若不是或不確定，移至 sbin/底下，在命令列輸入 **chmod 755 ***
- 完成以上步驟後, 在命令列輸入 **make MACHINE_TYPE=i686 World**
- 清除編譯過程中的暫存檔：**make cleanest**
- 安裝好後, 可在 bin/i686 中找到 SRILM 的執行檔

i686-m64:

- 64 bit machine (e.g., 資工系工作站)，請修改 **common/Makefile.machine.i686-m64**
  修改後執行 **make MACHINE_TYPE=i686-m64 World**
  安裝好後, 可在 bin/i686-m64 中找到 SRILM 的執行檔

Mac OS X: 如果系統是 Mac OS X，一開始的編譯方式會和上面不太一樣。

- 解壓縮 srilm-1.5.10.tar.gz
- 修改 Makefile，將 SRILM 路徑改成你要的路徑
- 修改 **common/Makefile.machine.macosx**
  line 13: 改為 CC = /usr/bin/gcc $(GCC_FLAGS) -Wimplicit-int
  line 14: 改為 CXX = /usr/bin/g++ $(GCC_FLAGS) -DINSTANTIATE_TEMPLATES
  line 49, 50: 註解掉
- 確定 srilm 安裝資料夾下的 sbin 為可執行檔
- 在 srilm 安裝資料夾下輸入 make MACHINE_TYPE=macosx World
- 清除暫存檔： make cleanest
- 之後會在 srilm 安裝資料夾 /bin/macosx 裡看到執行檔
- 要撰寫你自己 ngram_test 的 Makefile 時：
  MACHINE_TYPE = macosx
  CXX = /usr/bin/g++

Q2. SRILM compile error!

出現下述語句：

```
make: /home/master/98/r98922053/srilm-1.5.10/sbin/machine-type: Command not found
```

- 很有可能是因為你的系統中沒有安裝 csh
- 一般工作站應該都有安裝，若同學是使用自己的 linux 系統，可到套件管理程式那邊搜尋 csh 並安裝，
  或是在終端機輸入 **sudo apt-get install csh**

裝了 csh 還是出現以下 error:

```
make: /sbin/machine-type: Command not found
```

- 可能你的 SRILM 路徑含有中文，請確認是否為全英文路徑

```
matherr.c:18:16: warning: 'struct exception' declared inside parameter list will not be visible outside of this definition or declaration
matherr(struct exception *x)
     ^~~~~~~~~
matherr.c: In function 'matherr':
matherr.c:21:10: error: dereferencing pointer to incomplete type 'struct exception'
if (x->type == SING && strcmp(x->name, "log10") == 0) {
^~
matherr.c:21:20: error: 'SING' undeclared (first use in this function)
if (x->type == SING && strcmp(x->name, "log10") == 0) {
          ^~~~
matherr.c:21:20: note: each undeclared identifier is reported only once for each function it appears in
matherr.c:29:1: warning: control reaches end of non-void function [-Wreturn-type]
}
^
make[2]: *** [/home/master/98/r98922053/srilm-1.5.10/common/Makefile.common.targets:85: ../obj/i686-m64/matherr.o] Error 1
make[2]: Leaving directory '/home/master/98/r98922053/srilm-1.5.10/lm/src'
make[1]: *** [Makefile:77: release-libraries] Error 1
make[1]: Leaving directory '/home/master/98/r98922053/srilm-1.5.10'
make: *** [Makefile:51: World] Error 2
```

- 這是因為最新版的 glibc 2.27 已經把 struct exception 拿掉
- 將 srilm-1.5.10/lm/src/matherr.c 第 14 ~ 29 行註解掉即可

我使用 ubuntu 編譯 srilm，結果他回報 __off64_t 和 long int 的錯誤

- 這是 Ubuntu gcc library 自己的問題.....
- 可以修改 /usr/include/stdio.h
  把所有 __off64_t 取代成 long int 後儲存離開即可

Mac 執行 make MACHINE_TYPE=macosx World 後會出現以下 error message:

```
11 warnings and 1 error genertated.

make[2]: *** [../obj/macosx/LatticeIndex.o] Error 1
make[1]: *** [release-libraries] Error 1
make: *** [World] Error 2
```

- 請參考網頁 https://apurvtwr.wordpress.com/2012/11/08/installing-srilm-on-mac-10-8-2/
- 感謝 2014Autumn 修課的熊育萱同學

```
/usr/include/features.h:367:25 fatal error: sys/cdefs.h no such file or directory
```

- 可能因為電腦是 64bit 但卻用 32bit 去 compile

Q3. SRILM Makefile 前面的 # 是什麼?

代表註解掉 # 之後一行的意思，跟 C++ 中的 // 相同。因此在 srilm-1.5.10/Makefile 中應該要去掉 # 並設定 path：

- 把 **# SRILM = /home/speech/stolcke/project/srilm/devel**
  改成 **SRILM = 你的srilm資料夾** 即可，例如 **SRILM = /home/ntudsp2019fall/srilm-1.5.10**
- PS: 若想查詢完整路徑可在 srilm 的資料夾底下輸入 pwd 即可
- PS2: 因為檔案預設都是唯讀檔，你可以 **chmod 644 Makefile** 改變權限，或是在存檔時輸入 wq! 即可

Map Big5-ZhuYin.map to ZhuYin-Big5.map

Q4. mapping table 格式說明

- Character based 注音對應中文的 mapping 大概是長這樣:

```
ㄅ	八 匕 卜 不 卞 巴 比 丙 包 ...
八	八
匕	匕
不	不
...
...
ㄆ	仆 匹 片 丕 叵 平 扒 扑 疋 ...
仆	仆
匹	匹
片	片
...
...
ㄦ	二 而 耳 兒 洱 貳 爾 餌 邇 ...
二	二
而	而
兒	兒
...
...
```

  注意：每一個字之間都有空格
  實作上，列的順序不一定要按照字典排序。
- 注音文要轉換成字的時候，只取第一個注音轉換一個字（例如：ㄅ->罷），
  不是該字的所有注音都要能轉換（例如：不會有ㄚ->罷），也不會有多個注音轉到一個字（例如：ㄅㄨ->布）。
  注意：破音字可能會出現在多個不同注音作為開頭的行內。
- 不是所有注音都會出現在 map 中，例如沒有以ㄝ和ㄟ作為開頭的字，那麼ㄝ和ㄟ就不會出現在 map 中

Q5. mapping 產生 ZhuYin-Big5.map 可以用 python 寫嗎？

可以，但 mydisambig 必須要用 c++ 寫。關於使用 Makefile 執行 Python 程式可以參考 Makefile template 中 map 的部份。 使用 Python 讀檔與寫檔時，請用 'big5-hkscs' 作為 encoding。

Q6. Big5-ZhuYin.map 和 corpus.txt 開起來都是亂碼或問號？

- 因為這些檔案都是 Big5 編碼，若環境預設用 utf-8 或其他編碼去開，自然會出現亂碼。
- 若需要檢視檔案內容，可以使用 **iconv -f big5 $filename** 觀看，
  但需注意程式讀檔與寫檔均需使用 Big5 格式，請勿以其他編碼覆寫掉原本的檔案。

Using SRILM

Q7. 請問為何安裝之後無論我怎麼跑指令都不會產生任何結果也不會產生錯誤？

請注意全形半形的問題，不要直接把投影片上的指令 copy and paste 到 terminal 去執行

Q8. 在建立 lm 的時候, -unk 是處理什麼的 OOV？

在 testdata 中一定會有 corpus.txt 沒出現過的字，因此在 disambig 的時候就會變成 OOV。 在建立 lm 的時候加 -unk，打開 lm 就會發現有 <unk> 的機率，之後在 disambig 的時候就可以使用 <unk> 的機率代替 OOV 的機率。

Q9. 請問為何執行 **disambig -text testdata/xx.txt -map ZhuYin-Big5.map -lm bigram.lm -order 2 > $output**， 所有注音轉成了中文字，但是所有中文字都變 **<unk>** 了？

- ZhuYin-Big5.map 的格式是為了表示出 Viterbi 所需要的 obervation 與 state 之間的對應關係，
  也就是 tab 的左邊是 observation，右邊是所有可能的 state。
- 因此 ZhuYin-Big5.map 有兩種對應：注音(observation)對應到可能的字(state)，以及字(observation)對應到自己(state)。
- 所以當 **ZhuYin-Big5.map** 只有注音對應到字，沒有字對應到字，
  則雖然 disambig 讀入注音(observation)時，可以很順利的找出可能的字(state)，
  但讀入中文字(observation)時，由於 map 沒有對應的 state，因此會被視為 <unk>。
  或者 **testdata** 出現 **corpus.txt** 中沒有的字，也會因為 map 沒有對應的 state 而被視為 <unk>。
  因此請在執行時加添加 "-keep-unk"，也就是

disambig -text testdata/xx.txt -map ZhuYin-Big5.map -lm bigram.lm -order 2 -keep-unk > $output
這會使得 disambig 在遇到沒看過的字時，直接把輸入當成輸出。

Q10. 請問為何執行 **disambig -text testdata/xx.txt -map ZhuYin-Big5.map -lm bigram.lm -order 2 > $output**， 所有字都變 <unk> 了？

ZhuYin-Big5.map 的檔案格式應為 Big5，若為 utf-8 則可能使 disambig 將不同 encoding 的相同字當作是不同的，因此就會全部都是 <unk>。

檢查方法：在 terminal 打 **file ZhuYin-Big5.map**，只要是 ISO 開頭的應該就沒問題。

MyDisambig

Q11. How to include class Ngram and Vocab

- 如果同學要讀 SRILM 所訓練出來的 model，最簡單的方法就是 link SRILM 的 library。
  Compile 完之後，lib 的位置在 $(SRIPATH)/lib/$(MACHINE_TYPE)/ 之下，
  e.g. /root/srilm-1.5.10/bin/i686-m64。
- 如 source code 為 ngram_test.cpp，可撰寫 Makefile:

```
SRIPATH ?= /root/srilm-1.5.10
MACHINE_TYPE ?= i686-m64

CXX = g++
CXXFLAGS = -O3 -I$(SRIPATH)/include -w --std=c++11
vpath lib%.a $(SRIPATH)/lib/$(MACHINE_TYPE)

TARGET = ngram_test
SRC = ngram_test.cpp
OBJ = $(SRC:.cpp=.o)

.PHONY: all clean

all: $(TARGET)

$(TARGET): $(OBJ) -loolm -ldstruct -lmisc
	$(CXX) $(LDFLAGS) -o $@ $^

%.o: %.cpp
	$(CXX) $(CXXFLAGS) -c $<

clean:
	$(RM) $(OBJ) $(TARGET)
```

source code ngram_test.cpp 為:

```
#include <stdio.h>
#include "Ngram.h"

int main(int argc, char *argv[])
{

    int ngram_order = 3;
    Vocab voc;
    Ngram lm( voc, ngram_order );

    {
        const char lm_filename[] = "./corpus.lm";
        File lmFile( lm_filename, "r" );
        lm.read(lmFile);
        lmFile.close();
    }

    VocabIndex wid = voc.getIndex("囧");
    if(wid == Vocab_None) {
        printf("No word with wid = %d\n", wid);
        printf("where Vocab_None is %d\n", Vocab_None);
    }

    wid = voc.getIndex("患者");
    VocabIndex context[] = {voc.getIndex("癌") , voc.getIndex("毒"), Vocab_None};
    printf("log Prob(患者|毒-癌) = %f\n", lm.wordProb(wid, context));
}
```

如此就可以利用 lm.wordProb 來得到 language model 的機率。

- 示範檔下載: ngram_test.tar.gz (corpus.lm 非 char-based LM!!)
  使用前請先修改 Makefile 中的 SRIPATH 以及 MACHINE_TYPE!!

Q12. 什麼 function 可以測試字有沒有在字典裡面?

可以寫成下面這個樣子來測試

```
VocabIndex wid = voc.getIndex("囧");
if(wid == Vocab_None) {
    // replace OOV with <unk>
    wid = voc.getIndex(Vocab_Unknown);
}

// do something
...
```

Q13. 不太清楚 mydisambig 怎麼寫？

- bigram 和 trigram 的詳細推導
- 感謝 2018Spring 修課的林子雋同學